USL / DBMS NASA / PC R&D

WORKING PAPER SERIES

Report Number

DBMS.NASA/PC R&D-19

The USL/DBMS NASA/PC R&D Working Paper Series contains a collection of formal and informal reports representing results of PC-based research and development activities being conducted by the Center for Advanced Computer Studies of the University of Southwestern Louisiana pursuant to the specifications of National Aeronautics and Space Administration Contract Number NASW-3846 and NASA Training Grant Number NGT-19-010-900.

For more information, contact:

Wayne D. Dominick

Editor
USL/DBMS NASA/PC R&D Working Paper Series
Center for Advanced Computer Studies
University of Southwestern Louisiana
P. O. Box 44330
Lafayette, Louisiana 70504
(318) 231-6308

Object-Oriented Systems:

An Annotated Bibliography

Cherie T. Powell

and

Dennis R. Moreau

The University of Southwestern Louisiana
Center for Advanced Computer Studies
Lafayette, Louisiana

December 10, 1986

Object-Oriented Systems: An Annotated Bibliography

This document represents a comprehensive annotated bibliography of journal publications, conference publications and books related to object-oriented systems. This is an evolving document and will be updated periodically to reflect the current state of research literature in this area.

Abdali, S. Kamal, Guy W. Cherry and Neil Soiffer. "A Smalltalk System for Algebraic Manipulation." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 277-283.**

This paper describes the design of an algebraic computations system named VIEWS which has been written in Smalltalk. VIEWS has been designed to allow for the manipulation of a large variety of algebraic objects while still maintaining system cohesion and extensibility. The facilities developed to accomplish this include: the dynamic creation and manipulation of computational domains, viewing these domains as various categories (such as groups, rings, or fields), and expressing algorithms generically at the level of categories. These elements of the VIEWS system design led directly to the introduction of three new concepts added to Smalltalk: parameterized classes, protocol views, and categories. Overall, this provides a powerful basis for the design of a computer algebra system.

Agha, Gul A. "A Message-Passing Paradigm for Object Management." ***Database Engineering Bulletin,*** **Vol. 8, No. 4, December 1985, pp. 75-82.**

This article discusses the architecture of the actor model, which provides support for distributed databases. The author illustrates how all computations in the actor systems can be structured in terms of transactions. He also mentions Hybrid's implemented solutions for consistency, concurrency control, and deadlock.

Agha, Gul A. "An Overview of Actor Languages." ***ACM: Sigplan Notices,*** **October 1986, pp. 58-67.**

In this paper, information on the actor languages is presented with an overview of the actor model, plus a discussion on the advantages of actor languages in the design of large-scale concurrent architectures. Actors are agents that fulfill their actions when confronted by incoming communications. A discussion is also made on higher-level actor languages that use inheritance and delegation for conceptual organization and structuring respectively.

Ahlsen, Matts, Anders Bjornerstedt and Christer Hulten. "OPAL: An Object-Based System for Application Development." ***Database Engineering Bulletin,*** **Vol. 8, No. 4, December 1985, pp. 31-40.**

This paper presents an outline on OPAL, a combined run time and application development system. The authors also discuss the structure of the intended environment for OPAL, from the architecture to the basic system concepts including the Object-based model. A brief discussion is also presented on the development of an application within this model.

Anderson, David B. "Experience with Flamingo: A Distributed, Object-Oriented User Interface System." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 177-184.**

This paper deals with the experience of constructing and using a distributed object-oriented window manager. The goal of this experiment is to provide a flexible interface to a distributed, heterogeneous computing system. Through this object-oriented interface, Flamingo emerges as a kernel window manager which provides a dynamic linkage between different window management systems and graphic packages.

Atkinson, Robert G. "Hurricane: An Optimizing Compiler for Smalltalk." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 151-158.**

Hurricane is a recent and very efficient compilation system designed for the Smalltalk-80(TM) language. The unique aspect of this compiler is that it tries to make use the semantics of the language. This new approach can be handled through the development of a type declaration and inference mechanism. Details are provided for the implementation of the compiler.

Ballard, Mark B., David Maier and Allen Wirfs-Brock. "QUICKTALK: A Smalltalk-80 Dialect for Defining Primitive Methods." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 140-150.**

QUICKTALK is a dialect of Smalltalk-80 which was designed to depict the primitive Smalltalk methods. QUICKTALK provides better performance over bytecodes by abolishing the interpreter loop on bytecode execution, the removal of redundant class checking, and a decrease in the number of message send/returns via binding some target methods at compilation. This paper discusses the changes that were made in the Smalltalk-80 system and compiler.

Bhaskar, K. S., et al. "Virtual Instruments: Object Oriented Program Synthesis." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 303-314.**

Virtual Instruments is a support environment - implemented in Smalltalk-80 - which provides electronic test and measurement (T & M) applications. The main users are test engineers who specialize in the computer literate domain. In order to achieve software development without writing code, a programming paradigm composed of a bottom-up virtual instrument synthesis uses human interface models from the applications domain.

Black, Andrew, et al. "Object Structure in the Emerald System." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 78-86.**

A new object-based language for the construction of distributed applications is presented. Called Emerald, its goal is to simplify distributed programming through language support while also providing acceptable performance and flexibility, both locally and in the distributed environment. Emerald is fundamentally designed around objects which can encapsulate the concepts of process, procedure, data and location. In Emerald, objects are fully mobile and can move from node to node within the network, even during invocation. This paper emphasizes the various aspects of Emerald objects and the language's use of abstract types.

Bobrow, Daniel G., et al. "CommonLoops: Merging Lisp and Object-Oriented Programming." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 17-29.**

The authors describe the Common Lisp Object-Oriented Programming System, CommonLoops, which merges object-oriented programming smoothly into the procedure-oriented design of Lisp. By integrating Lisp data types with object classes, CommonLoops makes it easy to incrementally move a program between the procedure and object-oriented styles. One of the most important features of CommonLoops is the use of meta-objects which make practical both efficient implementation of and experimentation with new ideas for object-oriented programming. This system shares many similarities with several of the other important object-oriented languages and CommonLoops' small kernel is powerful enough to implement these languages.

Bonar, Jeffrey., Robert Cunningham and Jamie Schultz. "An Object-Oriented Architecture for Intelligent Tutoring Systems." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 269-276.**

This paper discusses an object-oriented architecture designed for a tutoring system which handles objects portraying knowledge elements that are taught by the tutors. These elements are known as bites which inherit both knowledge organization and tutoring components. The tutoring component provides the necessary features which handle tutoring tasks such as diagnosis, student modeling, and task selection. The authors demonstrate this approach through the implementation of several tutors.

Borgida, Alexander. "Exceptions in Object-Oriented Languages." *ACM: Sigplan Notices,* **October 1986, pp. 107-119.**

This paper describes how inflexible the object-oriented constraints are in the schema of conceptual model language. An example of this problem is to allow the existence of objects which violates constraints and to allow contradictory class definitions. The problems that exist with persistent exceptional objects can be resolved through a form of exception handling. Also featured are the advantages in building an exception handling language within an object-oriented paradigm.

Brown, Gretchen P., et al. "Program Visualization: Graphical Support for Software Development." *Computer,* **August 1985, pp. 27-35.**

Program Visualization is a graphical display which helps programmers form a precise mental image of a program's structure and function. The goal of the Program Visualization (PV) designers is to extend application-specific graphical support techniques and develop new graphics tools that together can be used at each stage of the software life cycle. After this process is completed, then the next challenge is to include the separate phases of the software development process into a unified conceptual framework.

Bruce, Kim B. and Peter Wegner. "An Algebraic Model of Subtypes in Object-Oriented Languages." *ACM: Sigplan Notices,* **October 1986, pp. 163-172.**

This paper illustrates how the definitions of type and subtype fail to correspond to their natural usage in programming languages. In an algebraic model, inheritance is expanded into a generalized version which means that the subtype relation is structured with property-preserving mappings. The algebraic model is also constructed from a second order (polymorphic) lambda calculus which contains subtype polymorphism which is related to inheritance.

Bruno, Giorgio and Alessandro Balsamo. "Petri Net-Based Object-Oriented Modelling of Distributed Systems." *OOPSLA'86: Sigplan Notices Special Issue,* **November 1986, pp. 284-292.**

This paper proposes the use of an object-oriented approach as a suitable means for building models of distributed systems. An example given is the case of Computer Integrated Manufacturing (CIM) Systems using ADA as the implementation language. A system using this approach is built up in three steps: control and synchronization aspects for each object class are treated using Petri nets, data are entered specifying the internal states of the objects and their passed messages, and the connections between the objects are introduced by a data flow diagram between classes. The approach offers advantages such as: support of conceptual models, integrated graphical representation, and building executable specifications. The flexibility of the approach and the possibility of using a knowledge-based user interface promote rapid prototyping and reusability.

Cargill, T. A. "Pi: A Case Study in Object-Oriented Programming." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 350-360.**

The subject of this paper is object-oriented programming's effect on the design, evolution and capability of a debugger called Pi. The author describes the development of Pi in C++ and why object-oriented programming worked well in that language. Object-oriented programming was initially chosen as a means of experimenting with a browser-type graphical user interface to the debugger. After the application of object-oriented techniques, several extra benefits such as symbol table structure, multi-process debugging and target environmental independence were also achieved.

Caudill, Patrick J. and Allen Wirfs-Brock. "A Third Generation Smalltalk-80(TM) Implementation." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 119-130.**

This paper concentrates on the design decisions which led to this new Smalltalk-80(TM) implementation. While Smalltalk-80(TM) retains features of its previous images, it also provides for a large object space and an interpreter. The interpreter has generation based garbage collection, but no object table. The performance results indicate that the Smalltalk-80(TM) implementation can handle a large number of active objects.

Cox, Brad and Bill Hunt. "Objects, Icons, and Software-IC's." ***BYTE: The Small Systems Journal,*** **August 1985, pp. 161-176.**

The paper discusses how object-oriented programming can make iconic user interface affordable. Iconic interface uses pictures instead of text and numbers as information guides, which increases the design cost. Encapsulation and inheritance are the two features which build the iconic interface through direct program integration of generic components and through the new application-specific components which inherit work from generic components in the library. The authors also illustrate a program called Workbench, which demonstrates how object-oriented programming and its ability to reuse code in generic Software-IC (reusable classes) libraries can build iconic applications at a reduced rate.

Croft, Bruce W. "Task Management for an Intelligent Interface." ***Database Engineering Bulletin,*** **Vol. 8, No. 4, December 1985, pp. 8-13.**

An intelligent interface assists users in the execution of their tasks. The paper discusses an intelligent interface that uses an object management system to manage object and task instantiations and their associations between one another. This object management system looks like a data model that emphasizes the modeling of operations.

Cunningham, Ward and Kent Beck. "A Diagram for Object -Oriented Programs." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 361-367.**

This paper introduces a system of notation for diagramming the message sending dialogue that takes place between objects that are involved in an object-oriented computation. In this system, objects are represented by boxes (labeled by the object's class) and messages are represented by directed arcs from the sending object to the receiving object. Other details of the notation system are also provided. The Smalltalk-80 system is used and its code has been extended to automatically collect information and construct diagrams utilizing an enhanced debugger utility. The authors have used these diagrams successfully as a teaching aid for students learning object-oriented programming.

Dasgupta, Partha. "A Probe-Based Monitoring Scheme for an Object-Oriented, Distributed Operating System." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 57-66.**

This paper investigates the use of probes for monitoring system status in a distributed system network. Probes are a form of emergency status enquiries that can be effectively used for system monitoring, reconfiguration, fault tolerance and interactive debugging support. A prototype operating/programming environment called Clouds, which is implemented on VAX/750 computers, is selected for probe usage. Clouds is a distributed, object-oriented operating system which is wholly structured on the object concept. The author presents a short summary of the design criteria, goals and architecture of the Clouds system. He also describes how the probe system fits into the existing Clouds design.

Decouchant, Dominique. "Design of a Distributed Object Manager for the Smalltalk-80 System." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 444-452.**

This presentation illustrates a distributed object manager which allows various Smalltalk-80 systems to share objects over a local-area network. The following are principles which support this object manager: location transparency and uniformed object naming, unique object representation and use of symbolic links for remote access, object migration and distributed garbage collection. An implemented version of this object manager is currently being integrated on a two nodes configuration.

De Jong, Peter. "Compilation into Actors." ***ACM: Sigplan Notices,*** **October 1986, pp. 68-77.**

This paper is about the ACT compiler which optimizes the translation of the Scripter programming language into actors. Details are given on the transformation of Scripter's synchronous function calls into asynchronous message sends which are acceptable to actors. Information is also provided on the conservation of time and space through the minimizing of the number of message sends and the number of customer and customer scripts generated.

Derrett, Nigel, William Kent and Peter Lyngbaek. "Some Aspects of Operations in an Object-Oriented Database." ***Database Engineering Bulletin,*** **Vol. 8, No. 4, December 1985, pp. 66-74.**

This paper discusses some aspects of database operations; specifically those for the Hewlett-Packard Iris DBMS prototype. The Iris data model is designed to be object-oriented by including data abstraction and the entity concept as central features. These features give the Iris object-oriented data model a clear advantage over a conventional data model (such as the Relational Model), by allowing database operations to be directly specified and stored in the DBMS. The authors go on to describe the specific operational features chosen for Iris and its prototype development.

Duff, Charles B. "Designing An Efficient Language." ***BYTE: The Small Systems Journal,*** **August 1985, pp. 211-224.**

The author discusses the design of Smalltalk and suggest ways to improve the language. Some of these improvements exist in garbage collecting, late versus early binding, and models for the interpreter. He also compares the advantages of a token-threaded interpreter to Smalltalk's byte-code interpreter in regards to a new language, called Actor, which supports early binding and other optimizations.

Ewing, Juanita J. "An Object-Oriented Operating System Interface." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 46-56.**

An object-oriented interface from a Smalltalk-80 programming environment to a UNIX-like operating system is presented and discussed. This interface imposes an object-oriented paradigm on operating system facilities. The goals for interface implementation include: access to low level operating facilities; use of higher order abstractions based on system calls, for operating system functions; cooperation between Smalltalk programs and conventionally compiled programs; creation of a window oriented operating system command interface. The author gives several examples of cooperating Smalltalk and operating system processes.

Fukunaga, Koichi and Shin-ichi Hirose. "An Experience with a Prolog-Based Object-Oriented Language." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 224-231.**

The object of this paper is to apply the SPOOL language to the annotated system PROMPTER. The SPOOL language is based on object-oriented and logic programming. The result of this combination was the formalization of domain knowledge into declarative data types which were reusable in different contexts. Further study is needed in the linguistic area to explore the full potential of this combination.

Gaguen, Joseph A. and Josi Mesegure. "Extensions and Foundations of Object-Oriented Programming." ***ACM: Sigplan Notices,*** **October 1986, pp. 153-162.**

This paper is about a new language, FOOPS (Foundations of Object-Oriented Programming System), which provides several new features, rigorous logical semantics, and unifies object-oriented programming with functional programming. FOOPS has three distinct techniques which are discussed: subsorts for multiple-inheritance, comparison between abstract data type and "object" modules, and a built-in data type written in a sub-language. Some of the new features that are mentioned are: user-definable abstract data type, flexible typing, semantics, and the mixing of coding with specifications.

Garrett, L. Nancy and Karen E. Smith. "Building a Timeline Editor from Prefab Parts: The Architecture of an Object-Oriented Application." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 202-213.**

Interval, a tool in the Intermedia system, allows inventors to explore new ideas and to build linked timelines. Interval is written in the object-oriented version of C. The purpose of this paper is to discuss the object-oriented architecture and the appropriate building blocks which makeup the main framework of the Interval system.

Hindler, James. "Enhancement for Multiple-Inheritance." *ACM: Sigplan Notices,* **October 1986, pp. 98-106.**

This presentation is about the improper usage of mixins within the multiple-inheritance class. Mixins is defined as the mixing of sets of operations in classes with usually larger classes. The paper presents us with a solution to this problem through an "enhancement" technique which has advantages over the present mixin method.

Ingalls, Daniel H. H. "A Simple Technique for Handling Multiple Polymorphism." *OOPSLA'86: Sigplan Notices Special Issue,* **November 1986, pp. 347-349.**

When a problem such as multiple polymorphic expressions exit in object-oriented programming, a breakdown occurs which causes the coding structure to disembark. A technique designed to solve this problem contains all of the object-oriented programming features which can confront any form of polymorphism. Although this technique is written in Smalltalk-80 syntax, other object-oriented languages can also be used.

Ishikawa, Yutaka and Mario Tokoro. "A Concurrent Object-Oriented Knowledge Representation Language Orient84/K: Its Features and Implementations." *OOPSLA'86: Sigplan Notices Special Issue,* **November 1986, pp. 232-241.**

This paper introduces an overview on the components of an object and its capability for concurrent execution in the Orient84/K language. A new schema is proposed and described for an adept execution of concurrent objects. A new interpreter, called ZOOM/VM, is designed to handle this new schema for Orient84/K.

Ishikawa, Yutaka and Mario Tokoro. "Concurrent Programming in Orient84/K: An Object-Oriented Knowledge Representation Language." *ACM: Sigplan Notices,* **October 1986, pp. 39-48.**

The object of this paper is to illustrate the design philosophy and concurrent constructs of Orient84/K programs. A description on each construct is given, such as asynchronous message passing, synchronization, priority control, access control, and mutual exclusion. An example program for mutual exclusion and access control demonstrates the capabilities of concurrent programming in Orient84/K.

Jacky, Jonathon and Ira Kalet. "An Object-Oriented Approach to a Large Scientific Application." *OOPSLA'86: Sigplan Notices Special Issue,* **November 1986, pp. 368-376.**

The authors describe their efforts to use object-oriented design in a large scientific application concerning the simulation of radiation therapy treatments for cancer. They describe how they instituted a practical clinical system with implemented objects, inheritance, message passing, windows, and concurrency, using Pascal. The system is implemented on a VAX minicomputer with graphical workstation capability, under a VMS operating system. The authors demonstrate the clear advantage of their object-oriented approach over the traditional scientific language (FORTRAN) array model.

Jacob, Robert J. K. "A State Transition Diagram Language for Visual Programming." ***Computer,*** **August 1985, pp. 51-59.**

The author discusses visual programming for abstract objects such as time sequence, hierarchy, conditional statements, and framed-based knowledge. A visual programming environment with a graphical language is presented to study the potential of visual programming. The author acknowledges that only a few graphical representations of abstract objects have been developed in the visual programming paradigm. Then he stresses the need for a more general graphical representation of programming objects.

Jacobson, Ivar. "Language Support for Changeable Large Real Time Systems." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 377-384.**

A set of concepts for the development and modeling of large telecommunication systems is presented. The original proposed study was to attempt to unify extensive experience in telecommunication systems design with emerging computer technology. Recent concepts growing out of this have resulted in an example object-oriented language called FDL that models large real time systems.

Johnson, Ralph E. "Type-Checking Smalltalk." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 315-321.**

The factor necessary to increase the speed of Smalltalk is called code optimization. The prerequisite required to build an optimized compiler for Smalltalk is a type-system. However, existing type-systems are not sufficient because of incorrect program typing, run-time type errors and not enough information for optimization. A new type-check system for Smalltalk which is fitting for code optimization is introduced.

Jones, Michael B. and Richard F. Rashid. "Mach and Matchmaker: Kernel and Language Support for Object-Oriented Distributed Systems." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 67-77.**

This paper describes Matchmaker, an interface specification language and compiler which provides programming language support for distributed, object-oriented programming. Its associated operating system is called Mach, a multiprocessor operating system kernel which provides interprocess communication based on capability. Their usage together provides for a heterogeneous, distributed, object-oriented programming environment. The authors discuss the development and operational performance of the Mach/Matchmaker environment and compare it to other related systems. Possible future directions are also examined.

Kaehler, Ted. "Virtual Memory on a Narrow Machine for an Object-Oriented Language." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 87-106.**

The author introduces a virtual memory called LOOM (Large Object-Oriented Programming) that is implemented in software that supports Smalltalk-80 on the Xerox Dorado computer. LOOM can provide secondary memory storage on narrow, 16-bit wide word computers. All storage is viewed as Smalltalk objects that contain fields. LOOM swaps objects between primary and secondary memory, and addresses each type of memory with a different sized object pointer. The author further discusses the design and implementation of LOOM as part of an integrated virtual memory and storage management system.

Kaehler, Ted and Dave Patterson. "A Small Taste of Smalltalk." ***BYTE: The Small Systems Journal,*** **August 1985, pp. 145-159.**

Through samples of various programs, the authors show how object-oriented programming can make a programmer's job simpler. A non-object-oriented program is presented as a learning tool for the Smalltalk language. Detailed information is also presented on the proper usage of the Smalltalk user interface. Finally, an example object-oriented program, implemented in Smalltalk, is written to demonstrate "The Animal Game".

Kahn, Kenneth, et al. "Objects in Concurrent Logic Programming Languages." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 242-257.**

Concurrent Prolog provides computational support for object-oriented programming. These support units are: incomplete messages, unification, direct broadcasting, and concurrency. Although Concurrent Prolog has an excellent support system, there are some doubts about Concurrent Prolog's support for expressing the abstractions of object-oriented programming. A preprocessor, known as Vulcan, can remedy this situation by exploring new variants of object-oriented programming which exists in this framework.

Kahn, Kenneth, et al. "Objects in Concurrent Logic Programming Languages." ***ACM: Sigplan Notices,*** **October 1986, pp. 29-38.**

Concurrent Prolog provides computational support for object-oriented programming. These support units are: incomplete messages, unification, direct broadcasting, and concurrency. Although Concurrent Prolog has an excellent support system, there are some doubts about Concurrent Prolog's support for expressing the abstractions of object-oriented programming. A preprocessor, known as Vulcan, can remedy this situation by exploring new variants of object-oriented programming which exists in this framework.

Khoshafian, Setrag N. and George P. Copeland. "Object Indentity." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 406-416.**

Identity is an element of an object that separates objects from other objects. The spectrum between weak and strong support of identity is examined. Plus, disagreements occur on why a strong identity should exist in the beginning of languages for general purpose programming and database systems. The authors also define a data model which describes complex objects and incorporates identity. A comparison between various implementation schemes for identity is presented. Debates on the purpose of a surrogate-based implementation schema, which supports a strong notion of identity, are also mentioned.

LaLonde, Wilf R., Dave A. Thomas and John R. Pugh. "An Exemplar Based Smalltalk." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 322-330.**

This paper discusses how exemplar-based systems are powerful in comparison to class based systems. Exemplar-based systems can distinguish between class hierarchies and instance hierarchies. The authors present the transformation of a class based Smalltalk to an exemplar-based Smalltalk. The authors also discuss the difference between or-inheritance and and-inheritance and they present an implementation of both.

Lang, Kevin J. and Barak A. Pearlmutter. "OAKLISP: An Object-Oriented Scheme with First Class Types." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 30-37.**

The authors introduce the reader to OAKLISP, a message based, multiple inheritance dialect of LISP. OAKLISP is based on a conceptual object-oriented language called Scheme. Using the Scheme philosophy, OAKLISP is designed to be simpler and more expressive than LISP by elevating functions to the level of first class objects which can be meaningfully manipulated by user code. Programs are written using LISP syntax, and traditional LISP data types coexist with a Smalltalk style class hierarchy. This provides OAKLISP with better integration of its object-oriented and functional sides than the other new object-oriented languages such as FLAVORS, Object LISP or COMMON LOOPS.

Lewis, David M., et al. "SWAMP: A Fast Processor for Smalltalk-80." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 131-139.**

SWAMP (Smalltalk Without All That Much Pipelining) is a processor designed to handle the rapid execution of Smalltalk-80 at a rate of 1.9M bytecode per second performance. This machine achieves this rapid execution through its circuitry, such as tag checking. The processor applies a general principle which locates special problems and handles them promptly.

Lieberman, Henry. "Using Prototypical Objects to Implement Shared Behavior in Object-Oriented Systems." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 214-223.**

This paper compares and contrasts the mechanisms of sets and prototypes for sharing behavior between objects in object-oriented programming languages. The author first covers the basic concepts of object representation with respect to artificial intelligence applications. He supports the prototype approach in AI as it corresponds more closely to the way people seem to acquire knowledge from concrete situations, whereas the concept of a set is more abstract and mathematical. The prototype approach holds some advantages for representing default knowledge, and incrementally and dynamically modifying concepts. The mechanism of delegation is used to implement this approach in object-oriented languages.

London, Ralph L. and Robert A. Duisberg. "Animating Programs Using Smalltalk." ***Computer,*** **August 1985, pp. 61-71.**

Animating programs provides information on how a program works through technical and psychological interfaces. Smalltalk is used in these animated programs because of its many facilities and human interface design. The authors discuss the many examples of animation programming through interactive compilers, application accelerators, distributed object managers, and dynamic programming language environments such as Prolog.

Maier, David, et al. "Development of an Object Oriented DBMS." *OOPSLA'86: Sigplan Notices Special Issue,* November 1986, pp. 472-482.

This paper presents the development of the GemStone object-oriented database server. The goals and requirements for the system are illustrated through the following subjects: an extensible data model featuring semantics, no artificial bounds on the size of database objects, database features and an interactive development environment. Smalltalk is used to answer some of these requirements, since GemStone supports a model of objects which are like Smalltalk's.

Maier, David, A. Otis and A. Purdy. "Object-Oriented Database Development at Servio Logic." *Database Engineering Bulletin,* Vol. 8, No. 4, December 1985, pp. 58-65.

GemStone is an object-oriented database system, which supports objects and messages similar to those in the Smalltalk-80 language. The authors discuss the requirements, goals, and the technical challenges that faced them during the development of this system.

McAllester, David And Ramin Zabih. "Boolean Classes." *OOPSLA'86: Sigplan Notices Special Issue,* November 1986, pp. 417-423.

Through the extension of classes any Boolean combination can exist as a class. Boolean classes allow a class of objects which governs a certain method to be properly named. Like predicates, a class can be declared true or false for any object. Although Boolean classes make classes more like predicates, they also preserve the inheritance hierarchy which exists at compile time.

McLeod, Dennis and Surjatini Widjojo. "Object Management and Sharing in Autonomous, Distributed Data/Knowledge Bases." *Database Engineering Bulletin,* Vol. 8, No. 4, December 1985, pp. 83-89.

A research project is implemented to devise and experimentally test concepts, techniques and mechanisms to support a distributed object management system. The system, known as the Distributed Personal Knowledge Manager (DPKM), would allow non-computer expert users to define, manipulate, and evolve collections of information. DPKM includes a set of primitive manipulation and retrieval operations, and a mechanism to allow controlled object sharing amoung multiple data/knowledge bases. A review of the development of a prototype implementation of this system is also provided.

Melamed, Benjamin and Robert J. T. Morris. "Visual Stimulation: The Performance Analysis Workstation." ***Computer*****, August 1985, pp. 87-94.**

This article presents an overview of work in the field of visual simulation and modeling. The visual method employs extensive graphics for drawing a model on a CRT screen and then observing its behavior through animation and dynamically evolving statistics. Some important advantages of this method include: increasing user efficiency by manipulating complex information pictorially instead of reading text, allowing the user visual access to the internal operations of the process, and letting the user experiment with his model by easily altering its structure and parameters and then observing changes at each step of the modified simulation. As part of their work on this subject, the authors have developed a new visual simulation package prototype called the Performance Analysis Workstation. It is written in C and runs in a UNIX-based environment. The paper describes the package development and experiences to date.

Meyer, Bertrand. "Genericity Versus Inheritance." ***OOPSLA'86: Sigplan Notices Special Issue*****, November 1986, pp. 391-405.**

This report compares and contrasts Genericity and Inheritance, which are two alternative methods for ensuring better extendibility, reusability and compatibility of software components. The author discusses a statically typed language, which was formed from the combined features of Genericity and Inheritance. He then presents the programming language Eiffel, which contains multiple inheritance and a limited form of genericity under full static typing.

Meyrowitz, Norman. "Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and Applications Framework." ***OOPSLA'86: Sigplan Notices Special Issue*****, November 1986, pp. 186-201.**

This paper describes the Intermedia system which is a large scale, object-oriented hypermedia system that provides document linkages. The author presents the educational and technological background within the system. Information is also given on the object-oriented technology that is used in the architecture and construction of the Intermedia system.

Miller, Michael S., et al. "The Application Accelerator Illustration System." ***OOPSLA'86: Sigplan Notices Special Issue*****, November 1986, pp. 294-302.**

An example of a CAD environment that supports the development of integrated circuits is the Application Accelerator Illustration System. Implemented in Smalltalk-80(TM), this system contains a hardware language, timing analyzer, functional simulator, waveform tracer, and data path place and route facility. The goal of the project is to integrate these aspects so that users can move freely with each feature of the system.

Mittal, Sanjay., Daniel G. Bobrow and Kenneth M. Kahn. "Virtual Copies: At the Boundary Between Classes and Instances." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 159-166.**

Object-oriented programming systems provide a good foundation for building knowledge bases by using networks of interconnected objects in their representations. In support of this, the authors describe a mechanism that provides a way to use a network as a prototype by making virtual copies of it. The use of virtual copies saves time and space in building knowledge bases for design, or for representing contexts in a problem solving system. The virtual copy mechanism extends the functions of object-oriented programming to deal with networks of objects rather than single instances, and allows multiple copies of networks of instances to be constructed incrementally.

Moon, David A. "Object Oriented Programming With Flavors." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 1-8.**

This paper describes an updated version of an object oriented programming system call Flavors. It was originally developed to build a window system and other system programming in a LISP program development environment. Flavors is a distributed approach to object oriented programming designed to encourage program modularity, easier development of large complex programs, and high run time efficiency. The author primarily focuses on the overall history, goals and major characteristics of the Flavors system.

Moriconi, Mark and Dwight F. Hare. "Visualizing Program Designs Through PegaSys." ***Computer,*** **August 1985, pp. 72-85.**

The PegaSys is an experimental system that explains program designs and encourages graphical images as formal, machine-processable documentation. The authors give background information on PegaSys and use excerpts to illustrate the style of interaction as well as the three main PegaSys capabilities.

Nierstrasz, O. M. "Hybrid: A Unified Object-Oriented System." ***Database Engineering Bulletin,*** **Vol. 8, No. 4, December 1985, pp. 49-57.**

The author discusses how an abstract data type language, Hybrid, tries to unify several object-oriented concepts into a single, coherent system. An overview of the object model and a number of the Hybrid language features are also mentioned. Information is also presented on object management.

Nguyen, V. and B. Hailpern. "A Generalized Object Model." ***ACM: Sigplan Notices,*** **October 1986, pp. 78-87.**

The authors present a new generalized object model that allows multi-dimensional inheritance. In this model, objects are organized into networks which eliminates the use of the large superclass and class of all classes. Objects, in this model, act like communicating processes which makes it possible to present simple yet formal semantics for objects and inheritance.

Nygaard, Kristen. "Basic Concepts in Object Oriented Programming." ***ACM: Sigplan Notices,*** **October 1986, pp. 128-132.**

This lecture is about the first object-oriented language, SIMULA. The author presents us with the basic foundation and characterization of object-oriented programming with respect to process and structure. In addition, the author views the prospects of object-oriented programming as a system which includes objects measured by their properties and by the changing of objects' states.

Olthoff, Walter G. "Augmentation of Object-Oriented Programming by Concepts of Abstract Data Type Theory: The ModPascal Experience." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 429-443.**

The subject of this report is the object-oriented language ModPascal, which is a part of the Integrated Software Development and Verification (ISDV) Project. The author shows how algebraic specification, enrichments, parameterized specification or signature morphism can be integrated into the ModPascal language. An investigation is also presented on how formal specification or algebraic verification can loose power and applications without the support of the above abstract data type (ADT) theoretical concepts.

Ossher, Harold L. "A Mechanism for Specifying the Structure of Large, Layered, Object-Oriented Programs." ***ACM: Sigplan Notices,*** **October 1986, pp. 143-152.**

This paper is about the clear concepts of the grid mechanism which is designed to enforce the structure of large, layered, object-oriented programs. The concepts include explicit identifications on program layers and specific access restrictions on program structures. Some techniques are also mentioned which make the global structure of large programs distinct.

Pascoe, Geoffrey A. "Elements of Object-Oriented Programming." ***BYTE: The Small Systems Journal,*** **August 1985, pp. 139-144.**

This paper first discusses comparative differences between procedure-oriented programming and object-oriented programming. In order for a language to be object-oriented, it must contain the following elements: information hiding, data abstraction, dynamic binding and inheritance. This article discusses these requirements and presents the advantages and disadvantages in using object-oriented programming languages.

Pascoe, Geoffrey A. "Encapsulators: A New Software Paradigm in Smalltalk-80." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 341-346.**

Encapsulators represents a new paradigm which builds structured and modular code in object-oriented systems. Encapsulators have applications to problems whose solutions can be cast in terms of performing automatic actions or enforcing access control when an object is sent a message. A discussion is made on how the new software paradigm can be properly integrated into the based Smalltalk-80 image.

Piersol, Kurt W. "Object Oriented Spreadsheets: The Analytic Spreadsheet Package." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 385-390.**

A spreadsheet that has been developed by Xerox for implementation in Smalltalk-80 is discussed. Called ASP, Analytic Spreadsheet Analysis, it has the capability to hold any type of Smalltalk-80 object in its spreadsheet cells giving it exceptional power, expandability and flexibility. The keys to ASP's capabilities lie in the options provided by Smalltalk such as the ability to treat programs as data, a readily available compiler, standardized protocols, and polymorphic variables.

Pountain, Dick. "Object-Oriented FORTH." ***BYTE: The Small Systems Journal,*** **August 1985, pp. 227-233.**

This article describes some FORTH extensions which, when compiled into a FORTH-83 standard system, form a new mechanism for defining abstract data systems which fits the description of real world objects. This new mechanism, implemented in FORTH, enables programmers to write object-oriented programs which are similar to Smalltalk-80 programs, but do not have the same power as Smalltalk.

Raeder, Georg. "A Survey of Current Graphical Programming Techniques." ***Computer,*** **August 1985, pp. 11-25.**

This article emphasizes how graphical images can improve current programming techniques. It discusses the interaction of the user with pictures and how this transfers information to the mind more rapidly than reading text. The most appropriate applications of pictures and images in programming are presented and some recent graphical-programming systems are reviewed.

Reiss, Steven P. "A Object-Oriented Framework for Graphical Programming." ***ACM: Sigplan Notices,*** **October 1986, pp. 49-57.**

GARDEN graphical programming system is a programming environment developed at Brown University. In this object-oriented framework, objects represent both programs and data. This paper describes how a practical environment based on graphical programming is achieved through display packages for objects and through a uniform exception mechanism that can handle the variations of graphical representations.

Samples, A. Dain, David Ungar and Paul Hilfinger. "SOAR: Smalltalk Without Bytecodes." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 106-118.**

This paper focuses on software techniques to support Smalltalk on conventional architectures. Smalltalk is implemented on an instruction level simulator for a reduced instruction set computer (RISC). The simulator is called SOAR (Smalltalk On A RISC) and is designed to improve the implementation and performance of Smalltalk-80 on a conventional computer. The primary change involves abandoning the virtual machine interpreter for Smalltalk-80, which has proved to be inherently slow in implementation, and instead compiling Smalltalk directly to SOAR machine code. Several changes in object and record manipulation had to be made for this new Smalltalk approach to work effectively.

Sandberg, David. "An Alternative to Subclassing." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 424-428.**

Smalltalk-80's expressive power is achieved through its class arrangements in a hierarchy. Although inheritance is the main feature of this hierarchy, description promotes the alternative organization of classes. The new option, called descriptive classes, not only uses compile-time type checking, but also the attributes of Smalltalk's hierarchy.

Schaffert, Craig, et al. "An Introduction to Trellis/Owl." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 9-16.**

This paper gives an overview of the design and development of a complete object-based programming environment called Trellis, and its implemented language, Trellis/Owl. This language provides for type (class) hierarchy with multiple inheritance which allows for shared operations. Trellis/Owl combines features that facilitate the design, implementation, and evolution of large, complex computer programs. The authors discuss the basic elements of the language and objects, and show how these are specified and implemented using concepts such as types, operations, and components.

Schmucker, Kurt J. "MACAPP: An Application Framework." ***BYTE: The Small Systems Journal,*** **August 1985, pp. 189-193.**

An application framework, called MacApp, is examined in this paper. The application framework defines the basic structures that implement the Macintosh user-interface standard without any specific commands. From this framework, a designer can customize his application through the design of object, object performance, and the installation of objects into the framework. The author presents the basic structure of the MacApp, suggestions on the development of an application, and how the MacApp can reduce a Macintosh program development time.

Schmucker, Kurt J. "Object-Oriented Languages for the Macintosh." ***BYTE: The Small Systems Journal,*** **August 1985, pp. 177-185.**

This paper gives an overview of the object-oriented languages which are used on the Apple Macintosh. A table is presented on the characteristics of each language, such as provisions for class methods and accessibility to the MacApp class library. Also provided is information on the mechanism of each programming language on the Macintosh.

Shu, Nan C. "FORMAL: A Forms-Oriented, Visual-Directed Application Development System." ***Computer,*** **August 1985, pp. 38-49.**

This article presents a forms-oriented programming language which simplifies the process of representing data objects and program structures. FORMAL is a practical forms-oriented language, which has a number of visual expressions that can compute a wide range of common data processing tasks. The author tries to prove the feasibility of FORMAL through a comparison between it and another two dimensional language.

Skarra, Andrea H. and Stanley B. Zdonik. "The Management of Changing Types in an Object-Oriented Database." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 483-495.**

This paper evaluates type evolution in a object-oriented database environment. Changing a type is not any easy task since types are defined as objects which are persistent and shared in a database environment. The extent of the change may exist in the object of the type and the programs that use objects of the type. In order for a type designer to create compatible versions of the type, a method is provided which supports timestamping and error handling mechanisms.

Smith, Reid G., Rick Dinitz and Paul Barth. "Impluse-86: A Substrate for Object-Oriented Interface Design." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 167-176.**

Impulse-86 is a foundation - implemented in the Strobe language - which provides the building blocks necessary to construct a variety of domain specific commands that support knowledge based systems. The five building blocks are: Editor, Editor Window, Property Display, Menu, and Operations. These building blocks can handle a range of interaction activities. The Impulse-86 substrate provides those developers - who are not familiar with interactive graphics - the ability to design a special interface for their system.

Snyder, Alan. "CommonObjects: An Overview." ***ACM: Sigplan Notices,*** **October 1986, pp. 19-28.**

CommonObjects is an object-oriented extension of Common Lisp which, in the style of Smalltalk and Flavors, supports object-oriented programming. In comparison to other languages, CommonObjects provides greater support to encapsulation with respect to inheritance. An efficient implementation strategy for conventional architectures is briefly described, plus an analysis on nonstandard language features which are necessary for portable implementations.

Snyder, Alan. "Encapsulation and Inheritance in Object-Oriented Programming Languages." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 38-45.**

The author presents an overview of two important concepts in object-oriented programming. The first one is encapsulation which is a technique to maximize modularity and data abstraction capability. The other concept is inheritance which allows subclasses access to all instance variables defined by an ancestor class. Unfortunately, in most object-oriented languages, the introduction of inheritance severely compromises the benefits of encapsulation. The paper discusses the encapsulation/inheritance relationship and develops requirements for full support of encapsulation with inheritance in object-oriented program systems.

Strom, Rob. "A Comparison of the Object-Oriented and Process Paradigms." ***ACM: Sigplan Notices,*** **October 1986, pp. 88-97.**

This paper compares the object-oriented and process paradigms with an emphasis on their development of large systems. Definitional differences are presented on particular instances of the paradigms. The mechanisms of each paradigm needed to support dynamic code binding, code reuse, and access control are presented and contrasted.

Stroustrup, Bjarme. "An Overveiw of C++." ***ACM: Sigplan Notices,*** **October 1986, pp. 7-18.**

C++ is an added version of the C programming language. C++ was designed to incorporate new features into the C language which can handle type checking, data abstractions, operation overloading, and object-oriented programming.

Tesler, Larry. "Programming Experiences." ***BYTE: The Small Systems Journal,*** **August 1985, pp. 195-206.**

The author compares the experience of different object-oriented programming designers. He interviewed several people who designed their programs in Objective-C, C++, Object Pascal, and the Smalltalk language. He collected each programmer's thoughts about object-oriented programming and its effect on their projects. These interviews prove how diversifiable these object-oriented programming languages can be for various situations.

Tsichritzis, D. "Object Species." ***Database Engineering Bulletin,*** **Vol. 8, No. 4, December 1985, pp. 2-5.**

This paper discusses an environment for end-user-oriented objects, which can be used in an Office Information System and in particular a Message System. The author studies the specification and implementation of complex objects which have an external behavior that can be visualized by users through analogies from the animal world.

Vegdahl, Steven R. "Moving Structures Between Smalltalk Images." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 466-471.**

Moving data structures between Smalltalk images is the subject of this paper. Existing Smalltalk images do not have the necessary facilities to handle circular structures properly. A collection of Smalltalk methods, which has advantages over the standard image, was implemented to handle circular structures. The issues that arose during the design, implementation, and usage of these methods were discussed in this paper.

Wegner, Peter. "Classification in Object-Oriented Systems." ***ACM: Sigplan Notices,*** **October 1986, pp. 173-182.**

This paper reviews the classification mechanisms which categorize object-oriented systems. A brief discussion is presented on the contrast between communication and state transition paradigms. There is also a focus on how type inheritance transforms from simple descriptive categories to tree-structured hierarchies. Technical details are also included on the structures of algebraic, second order, polymorphic, and lambda calculus in regards to values that are computed with a calculus of classes weaker than calculi.

Weiser, Stephen P. "An Object-Oriented Protocol for Managing Data." ***Database Engineering Bulletin,*** **Vol. 8, No. 4, December 1985, pp. 41-48.**

The author presents an object-oriented programming environment, named Oz, which was designed to easily handle certain aspects of office information systems. Also mentioned are the limitations of Oz and the additional facilities which enhance Oz's capability of managing office data.

Wiebe, Douglas. "A Distributed Repository for Immutable Persistent Objects." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 453-465.**

Jasmine is an object-oriented system for programming-in-the-large which uses "system model objects" to influence software development. These objects are lifelong and unchangeable since the system models act like historical records. Information is also presented on the JStore, which is a storage system that provides robust, transitional, and write-once storage. The paper further describes the designs for the serialization, location and replication of objects that are used in JStore.

Woo, C. C. and F. H. Lochovsky. "An Object-Based Approach to Modelling Office Work." ***Database Engineering Bulletin,*** **Vol. 8, No. 4, December 1985, pp. 14-22.**

This paper discusses an object system designed to support office work. The authors examine the necessary facilities that must exist in "objects" in order to support different types of office work. They also demonstrate this new model through an office example.

Yokote, Yasuhiko and Mario Tokoro. "The Design and Implementation of ConcurrentSmalltalk." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 331-340.**

ConcurrentSmalltalk is a programming language/system which is outlined in terms of communication, synchronization, atomic objects, and concurrent constructs. The design and implementation of ConcurrentSmalltalk is also discussed through details on bytecodes, primitives, object representation and garbage collection.

Yonezawa, Akinori, Jean-Pierre Briot and Etsuya Shibayama. "Object-Oriented Concurrent Programming in ABCL/1." ***OOPSLA'86: Sigplan Notices Special Issue,*** **November 1986, pp. 258-268.**

An object-oriented computation model, which incorporates three types of message passing, is designed to demonstrate a variety of concurrent systems. ABCL/1 is the programming language that supports this computation model and through its use, an implementation of distributed problem solving is presented. An example of the reply destination mechanism and future type message passing is given by a distributed "same fringe" algorithm which compares the fringes of two trees (LISP list).

Zdonik, Stanley B. "Maintaining Consistency in a Database with Changing Types." ***ACM: Sigplan Notices,*** **October 1986, pp. 120-127.**

Upholding conformity between a set of persistent objects and a set of changeable type definitions is a difficult problem that exists in object-oriented database systems. The author demonstrates the solution through a set of error handlers connected with versions of a type and a version control mechanism.

Zdonik, Stanley B. "Object Management Systems for Design Environments." ***Database Engineering Bulletin,*** **Vol. 8, No. 4, December 1985, pp. 23-30.**

This paper also discusses an object system designed to support office work.

5.19

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
111-82	~~44736~~ 183589	

4. Title and Subtitle	5. Report Date
USL/NGT-19-010-900: OBJECT-ORIENTED SYSTEMS: AN ANNOTATED BIBLIOGRAPHY	December 10, 1986 DATE OVERRIDE
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
CHERIE T. POWELL AND DENNIS R. MOREAU	
	10. Work Unit No.

9. Performing Organization Name and Address	11. Contract or Grant No.
University of Southwestern Louisiana The Center for Advanced Computer Studies P.O. Box 44330 Lafayette, LA 70504-4330	NGT-19-010-900
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	FINAL; 07/01/85 - 12/31/87
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This Working Paper Series entry represents a comprehensive annotated bibliography of journal publications, conference publications, and books related to object-oriented systems. This is an evolving document and will be updated periodically to reflect the current state of research literature in this area.

This report represents one of the 72 attachment reports to the University of Southwestern Louisiana's Final Report on NASA Grant NGT-19-010-900. Accordingly, appropriate care should be taken in using this report out of the context of the full Final Report.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Object-Oriented Systems Bibliography, PC-Based Research and Development	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	26	

*For sale by the National Technical Information Service, Springfield, Virginia 22161